

Advice Complexity of the Online Search Problem

Jhoirene Clemente^{1*}, Juraj Hromkovič^{2**}, Dennis Komm², and Christian Kudahl^{3***}

¹ Department of Computer Science,
University of the Philippines Diliman, Philippines,
jbclemente@up.edu.ph

² Department of Computer Science,
ETH Zürich, Switzerland,

{juraj.hromkovic,dennis.komm}@inf.ethz.ch

³ Department of Mathematics and Computer Science,
University of Southern Denmark, Denmark,
kudahl@imada.sdu.dk

Abstract. The online search problem is a fundamental problem in finance. The numerous direct applications include searching for optimal prices for commodity trading and trading foreign currencies. In this paper, we analyze the advice complexity of this problem. In particular, we are interested in identifying the minimum amount of information needed in order to achieve a certain competitive ratio. We design an algorithm that reads b bits of advice and achieves a competitive ratio of $(M/m)^{1/(2^b+1)}$ where M and m are the maximum and minimum price in the input. We also give a matching lower bound. Furthermore, we compare the power of advice and randomization for this problem.

1 Introduction

We study the online search problem (abbreviated ONLINE SEARCH), which is formulated as an online (profit) maximization problem. For such problems, the input arrives gradually in consecutive time steps. Each piece of input is called a *request*. After a request is given, an online algorithm (also called the *online player*) has to produce a definite piece of the output, called an *answer*. Each answer is thus computed without any knowledge about further requests [6]. The goal is to produce an output with a *profit* that is as large as possible. In ONLINE SEARCH, the online player searches for the maximum price of a certain asset that unfolds sequentially. Suppose the player, in this context a trader, would like to transfer its assets from, say, USD to CHF in one transaction. Each day (formally, each time step), the trader receives a quotation of the current exchange rate and

* Supported by ERDT Scholarship. Sandwich program funded by PCIEERD-BCDA.

** Supported by SNF grant 200021-146372.

*** Supported by the Villum Foundation and the Stibo-Foundation.

decides whether to trade on the same day or to wait. The trading duration is finite, and it may be known or unknown to the trader. Formally, we define ONLINE SEARCH as follows.

Definition 1 (Online Search Problem). *Let $\sigma = (p_1, p_2, \dots, p_n)$, with $0 < m \leq p_i \leq M$ for all $1 \leq i \leq n$, be a sequence of prices that arrives in an online fashion. Here, M and m are upper and lower bounds on the prices, respectively. For each day i , price p_i is revealed, and the online player has to choose whether to trade on the same day or to wait for the new price quotation on the next day. If the player trades on day i , its profit is p_i . If the player did not trade for the first $n - 1$ days, it must accept p_n . The player's goal is to maximize the obtained price (i. e., its profit).*

We assume that the parameters m and M for the price range are fixed and known to the online algorithm in advance. The duration of the trading period n is finite and may or may not be known to the online algorithm. We do not take into account sampling costs in the profit, i. e., the price for each day is freely given by the market to the trader. However, some direct applications of ONLINE SEARCH may do require to consider the sampling costs. For instance, obtaining prices of a certain product may induce some cost, either in the form of time or money, from the player. For a study of such more involved cost variants, we refer the reader to Xu et al. [20], where the authors considered the accumulated sampling cost while maximizing the player's profit.

1.1 Competitive Analysis and Advice Complexity

Competitive analysis was introduced by Sleator and Tarjan in 1985 [18] to analyze the solution quality of online algorithms. The measure used in the analysis is called the *competitive ratio*, which can be obtained by comparing the profit of the online algorithm to the one of an optimal offline solution. The term “offline” is used when the whole input sequence is known in advance. Note that it is generally not possible for an online algorithm to compute the optimal offline solution in advance, because parts of the output have to be specified before the whole input is known. It is merely taken into account to analyze the profit that can hypothetically be obtained if the whole input is known in advance. The competitive ratio of an online algorithm is formally defined as follows.

Definition 2 (Competitive Ratio). *Let Π be an online maximization problem, let ALG be an online algorithm for Π , and let $c > 1$. ALG is said to be c -competitive if, for every instance I of Π , we have*

$$c \cdot \text{profit}(\text{ALG}(I)) \geq \text{profit}(\text{OPT}(I)) ,$$

where $\text{profit}(\text{ALG}(I))$ is the profit of ALG on input I , and $\text{profit}(\text{OPT}(I))$ denotes the optimal offline profit.

In this paper, we study the *advice complexity* of ONLINE SEARCH. More specifically, we ask about the additional information both sufficient and necessary in order to improve the obtainable competitive ratio. In a way, this approach can be seen as measuring the *information content* of the problem at hand [13].

This tool, which was introduced by Dobrev et al. in 2008 [9] and then revised by Böckenhauer et al. [4], Hromkovič et al. [13], and Emek et al. [11], is a complementary tool to analyze online problems. In order to study the information that is needed in order to outperform purely deterministic (or randomized) online algorithms, we introduce a trusted source, referred to as an *oracle*, which sees the whole input in advance and may write binary information on a so-called *advice tape*. These *advice bits* are allowed to be any function of the entire input. The algorithm, which is called an *online algorithm with advice* in this setting, may then use the advice to compute the output for the given input. The approach is quantitative and problem-independent. In other words, the information supplied can be arbitrary (as long as it is computable). This is in particular interesting to give lower bounds for many other measurements or relaxations of online problems. More specifically, hardness results in advice complexity give useful negative results about various semi-online approaches. If it is for example shown that $O(\log_2 n)$ bits of advice do not help any online algorithm to achieve a better competitive ratio, this gives a negative answer to questions of the form: Would it help the algorithm to know the length of the input? Would it help the algorithm to know the number of requests of a certain type?

Many prominent online problems have been studied in this framework, including paging [9,4], the k -server problem [4,11,12,17], metrical task systems [11], and the online knapsack problem [5]. Negative results on the advice complexity can be transferred by a special kind of reduction [2,8,11]. Moreover, advice complexity has a close and non-trivial relation to randomization [3,14,16]. We now define online algorithms with advice formally.

Definition 3 (Advice Complexity). *Let x_1, \dots, x_n be the input for an online problem Π . An online algorithm with advice, ALG, for Π computes the output sequence y_1, \dots, y_n , where y_i is allowed to depend on x_1, \dots, x_{i-1} as well as on an advice string ϕ . The advice, ϕ , is written in binary on an infinite tape and is allowed to depend on the request sequence x_1, \dots, x_n . The advice complexity of ALG is the largest number of advice bits it reads from ϕ over all inputs of length at most n .*

Our paper is devoted to both creating online algorithms with advice for ONLINE SEARCH that achieve a certain output quality while using a certain number of advice bits, and to show that such algorithms cannot exist if the advice complexity is below some certain threshold.

Most of the work in advice complexity theory considers problems where at least n advice bits are required for an algorithm to be optimal. Here, we study a problem where only $\log_2 n$ bits give an optimal algorithm. We investigate how this problem behaves when the number of advice bits is in the interval $[1, \log_2 n]$. For the ease of presentation, we assume that $\log_2 n$ is integer.

2 Related Work

The search problem in an offline setting, i. e., where the set of prices is known in advance, can easily be solved optimally in time $O(n)$. However, for a lot of online environments such as stock trading and foreign exchange, decisions should be made even though there is no knowledge of the future prices of the currencies. These problems are intrinsically online.

The most common approaches are Bayesian. These approaches rely on a prior distribution of prices where the online algorithm computes a certain *reservation price* based on the distribution. The trader accepts any price that is larger than or equal to the reservation price. If this certain price is not met, the player has to trade on the last day (according to Definition 1). Throughout this paper, $\text{ALG}[p]$ denotes the algorithm that accepts the first price it sees that is at least p .

Since the prior distribution of prices is not necessarily known in advance, El-Yaniv et al. [10] proposed to measure the quality of online trading algorithms using competitive analysis. Moreover, for some assets, the goal is not just to increase the profit but to minimize the loss by considering the possible worst-case scenarios in the market. Competitive analysis in financial problems such as ONLINE SEARCH can provide a guaranteed performance measure for the trader's profit. The best deterministic online algorithm with respect to competitive analysis is $\text{ALG}[\sqrt{Mm}]$, i. e., the algorithm that accepts the first price it sees that is at least \sqrt{Mm} (or it accepts p_n if no such price is ever seen). This algorithm has a competitive ratio of $\sqrt{M/m}$, which is provably the best competitive ratio any deterministic online algorithm without advice can achieve [6].

Boyar et al. [7] studied how the problem behaves when applying a variety of difference performance measures (and not just competitive ratio).

3 Advice for the Online Search Problem

In this section, we explore the advice complexity of ONLINE SEARCH. We start by studying how much advice is necessary and sufficient in order to obtain an optimal output. After that, we study general c -competitiveness.

3.1 Advice for Optimality

It is possible for an algorithm to be optimal using $\log_2 n$ bits of advice if n is known in advance by simply encoding the day where the largest price is offered. If n is not known in advance, it has to be encoded with a self-delimiting encoding, for example, by writing the length of $\log_2 n$ in unary followed by $\log_2 n$. This requires $2 \log_2 n$ bits [4].

Moreover, optimality can also be achieved by encoding the value of p_{\max} using $O(\log_2(M/m))$ bits, but since M and m can be arbitrarily large, this may be very expensive. We now give a complementing lower bound.

Theorem 1. *At least $\log_2 n$ bits of advice are necessary to obtain an optimal solution for ONLINE SEARCH. This holds even if n is known to the algorithm.*

Proof. We use that an algorithm with b advice bits can be viewed as dealing with the best of 2^b algorithms without advice, for the particular instance chosen. First, we generate a set of request sequences \mathcal{S} . Then, we show that, for \mathcal{S} , there is no set of $n - 1$ or fewer deterministic algorithms, which can ensure that at least one algorithm always gets the optimal solution.

We construct the set \mathcal{S} in such a way that each request has a unique optimal solution. The construction is as follows. Let $\mathcal{S} = \{\sigma_1, \sigma_2, \dots, \sigma_n\}$, such that

$$\sigma_i = (\underbrace{m + \delta, m + 2\delta, \dots, m + i\delta}_i, \underbrace{m, \dots, m}_{n-i}),$$

where $\delta = (M - m)/n$. Each σ_i is thus a sequence of n prices that follow an increasing order until day i . Then the price drops to the minimum m for the remaining $n - i$ days. The optimal solution for each σ_i clearly is to trade on day i and obtain a profit of $m + i\delta$. From the construction, it is impossible for any deterministic online algorithm to distinguish the request sequence σ_i from any other sequence of requests σ_j , for $j > i$, until the price for day $i + 1$ is offered. This is due to the fact that the set of requests $\{\sigma_i, \sigma_{i+1}, \dots, \sigma_n\}$ have the same prices offered from day 1 up to day i . Since we have n such input instances with different optimal solutions, and fewer than n algorithms, there is one algorithm that gets chosen for at least one of the above instances. Clearly, this algorithm cannot be optimal for both these instances. Thus, any online algorithm with advice needs $\log_2 n$ bits of advice to identify the actual input from these n possible cases. \square

Note that if it is required that the prices are integral, this construction still works by picking m and M such that δ is an integer.

3.2 Advice for c -Competitiveness

Next, we investigate the advice complexity of ONLINE SEARCH if we have less than $\log_2 n$ advice bits. This means we study a tradeoff between the number b of advice bits supplied and the competitive ratio c obtainable. Recall that, without advice bits, the optimal trader strategy is to use $\text{ALG}[p]$, where the reservation price is $p = \sqrt{Mm}$.

Before we present the upper bounds for online algorithms with advice for ONLINE SEARCH that achieve c -competitiveness, we give a simple intuition behind our strategy. We can think of it as having 2^b deterministic algorithms with different reservation prices. The computation of each reservation price p_i is obtained by computing the solution of the following equation.

$$\frac{p_1}{m} = \frac{p_2}{p_1} = \dots = \frac{p_{2^b}}{p_{2^b-1}} = \dots = \frac{M}{p_{2^b}}$$

Theorem 2. *For every $b > 0$, there exists an online algorithm with advice for ONLINE SEARCH which reads b bits of advice and achieves a competitive ratio of at most $(M/m)^{\frac{1}{2^b+1}}$. This holds even if n is unknown.*

Proof. We describe an algorithm ALG with advice which reads b bits of advice and achieves the claimed competitive ratio. First, the oracle simulates the algorithms

$$\text{ALG} \left[m^{\frac{2^b+1-i}{2^b+1}} M^{\frac{i}{2^b+1}} \right]$$

for $i = 1, \dots, 2^b$. Let A denote the set of these algorithms. Then, it writes the value of i for the algorithm that achieves the best competitive ratio. We argue that at least one of the algorithms gets a competitive ratio of at most

$$\left(\frac{M}{m} \right)^{\frac{1}{2^b+1}}.$$

We have three cases for p_{\max} . The first case is when $p_{\max} < m^{\frac{2^b}{2^b+1}} M^{\frac{1}{2^b+1}}$. Here, each algorithm in A will get the price offered on the last day, which is at least m . The competitive ratio for ALG is at most

$$\frac{m^{\frac{2^b}{2^b+1}} M^{\frac{1}{2^b+1}}}{m} = \left(\frac{M}{m} \right)^{\frac{1}{2^b+1}}.$$

The second case is when $p_{\max} \geq m^{\frac{1}{2^b+1}} M^{\frac{2^b}{2^b+1}}$. In this case,

$$\text{ALG} \left[m^{\frac{1}{2^b+1}} M^{\frac{2^b}{2^b+1}} \right]$$

gets a price of at least $m^{\frac{1}{2^b+1}} M^{\frac{2^b}{2^b+1}}$. Since OPT gets at most M , the competitive ratio for ALG is again at most

$$\frac{M}{m^{\frac{1}{2^b+1}} M^{\frac{2^b}{2^b+1}}} = \left(\frac{M}{m} \right)^{\frac{1}{2^b+1}}.$$

The last case is when $m^{\frac{2^b+1-i}{2^b+1}} M^{\frac{i}{2^b+1}} \leq p_{\max} < m^{\frac{2^b-i}{2^b+1}} M^{\frac{i+1}{2^b+1}}$ for some $i < 2^b$. In this case,

$$\text{ALG} \left[m^{\frac{2^b+1-i}{2^b+1}} M^{\frac{i}{2^b+1}} \right]$$

gets at least its reservation price. Thus, also here, the competitive ratio for ALG is at most

$$\frac{m^{\frac{2^b-i}{2^b+1}} M^{\frac{i+1}{2^b+1}}}{m^{\frac{2^b+1-i}{2^b+1}} M^{\frac{i}{2^b+1}}} = \left(\frac{M}{m} \right)^{\frac{1}{2^b+1}}.$$

All in all, we have shown that, in each case, ALG obtains a competitive ratio of at most $(M/m)^{\frac{1}{2^b+1}}$ as we claimed. \square

We now present a matching lower bound.

Theorem 3. *Let ALG be an algorithm with advice for ONLINE SEARCH which reads $b < \log_2 n$ bits of advice. The competitive ratio of ALG is at least $(M/m)^{\frac{1}{2^b+1}}$.*

Proof. For any given $b < \log_2 n$, let ALG be an algorithm with advice that reads at most b bits of advice. Again, we view this advice as 2^b deterministic online algorithms. We now give a class of request sequences that ensure that each of them gets a competitive ratio of at least

$$\left(\frac{M}{m}\right)^{\frac{1}{2^b+1}}.$$

Consider the sequence $(p_1, p_2, \dots, p_{2^b})$ with

$$p_i = m^{\frac{2^b+1-i}{2^b+1}} M^{\frac{i}{2^b+1}}.$$

The adversary simulates all 2^b algorithms on this sequence. We consider two cases. If a request p_i is rejected by all algorithms, it requests p_1, p_2, \dots, p_i followed by requests that are all equal to m . For the first case, assume that there exists a request p_i , which is rejected by all 2^b algorithms. The remaining requests are all m . This means that ALG gets a price of at most p_{i-1} (the largest request that was not p_i) while OPT gets a price of p_i . Note that, if the first request is rejected, ALG gets a price of at most $m = p_0$. In this case, the competitive ratio for ALG is at least

$$\frac{p_i}{p_{i-1}} = \frac{m^{\frac{2^b+1-i}{2^b+1}} M^{\frac{i}{2^b+1}}}{m^{\frac{2^b+2-i}{2^b+1}} M^{\frac{i-1}{2^b+1}}} = \left(\frac{M}{m}\right)^{\frac{1}{2^b+1}}.$$

Thus, ALG cannot obtain a competitive ratio which is better than $(M/m)^{\frac{1}{2^b+1}}$ if a request is rejected by all the algorithms.

Next, we consider the second case. Here, every request in σ is accepted by some algorithm. Since there are 2^b requests in σ , it follows that all algorithms accept a price that is at most p_{2^b} . Since $2^b < n$, the adversary can still make a request. The final request is then M . The competitive ratio for ALG is therefore bounded from below by

$$\frac{M}{p_{2^b}} = \frac{M}{m^{\frac{1}{2^b+1}} M^{\frac{2^b}{2^b+1}}} = \left(\frac{M}{m}\right)^{\frac{1}{2^b+1}}.$$

In both cases, ALG has a competitive ratio of at least $(M/m)^{\frac{1}{2^b+1}}$ as claimed by the theorem. \square

4 Advice and Randomization

Randomization is often used to improve the competitive ratio of online algorithms (in expectation). Here, the online player is allowed to base some of its

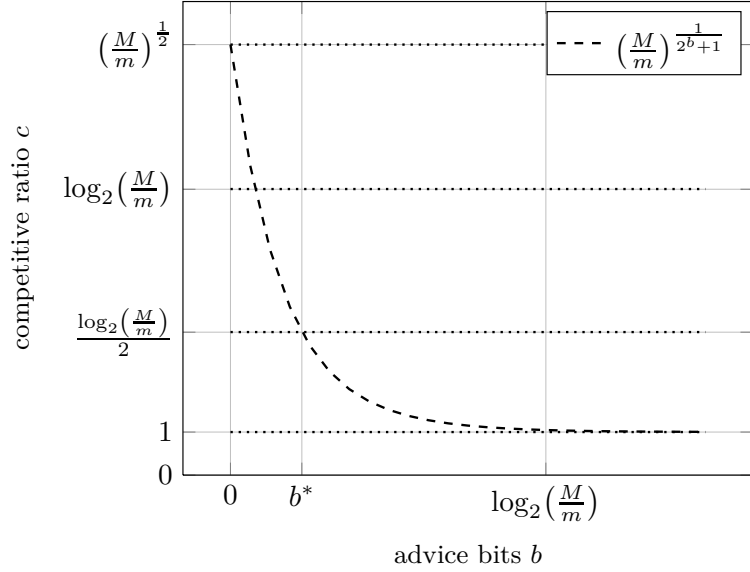


Fig. 1. Plot comparing the competitive ratio of the online algorithm with advice with respect to the lower bound for deterministic and randomized algorithms.

answers on a random source. An oblivious adversary knows the algorithm, but not the outcome of the random decisions. To provide an improvement over the lower bound of deterministic online algorithms for ONLINE SEARCH, El-Yaniv et al. [10] provided an upper bound by presenting a randomized algorithm with an expected competitive ratio of $\log_2(M/m)$. Lorenz et al. [15] provided an asymptotically matching lower bound of $(\log_2(M/m))/2$ for randomized online algorithms for ONLINE SEARCH.

In this section, we compare the power of advice to the ability of an online algorithm to access random bits for ONLINE SEARCH. The competitive ratio of online algorithms with advice (with an increasing number of advice bits) is shown in Figure 1. We fixed a fluctuation ratio M/m , and we highlighted the competitive ratio of the best deterministic algorithm, i.e., $(M/m)^{\frac{1}{2}}$, and the corresponding upper (i.e., $\log_2(M/m)$) and lower (i.e., $\log_2(M/m)/2$) bounds of randomized algorithms for ONLINE SEARCH.

It is interesting to point out that, with the number of advice bits greater than

$$b^* = \log_2 \left(\frac{\log_2(M/m)}{\log_2\left(\frac{\log_2(M/m)}{2}\right)} - 1 \right),$$

our online algorithm for ONLINE SEARCH outperforms the lower bound of randomized online algorithms. And as we increase the number of advice bits, the better the competitive ratio we get. In the plot shown in Figure 1, we considered

a fluctuation ratio $M/m = n$. Note that the competitive ratio is asymptotic to 1, but it is actually possible to get an optimal solution with $\log_2 n$ advice bits.

5 Conclusion and Future Work

We studied the advice complexity of **ONLINE SEARCH** and determined upper and lower bounds on the advice complexity to achieve both optimality and c -competitiveness. We presented a tight lower bound of $\log_2 n$ for the number of advice needed by any online algorithm to obtain optimal solutions, as shown in Theorem 1. We also provided a strategy with b bits of advice and achieved a tight bound of $(M/m)^{\frac{1}{2^b+1}}$ for the competitive ratio as shown in Theorems 2 and 3.

We compared the power of advice and randomization in terms of competitive ratio. The comparison of the competitive ratio is shown in Figure 1.

For future work, it would be interesting to extend the results to the **ONE-WAY TRADING** problem with advice. It is known that **ONLINE SEARCH** and the **ONE-WAY TRADING** are closely related. In fact, they are equivalent in the sense that, for every randomized algorithm for **ONLINE SEARCH**, there exists an equivalent deterministic algorithm for **ONE-WAY TRADING** [10]. Although randomization significantly improved the competitive ratio of algorithms for **ONLINE SEARCH**, it can be shown that it cannot help to improve the competitive ratio of algorithms for **ONE-WAY TRADING**. It would be interesting to investigate the tradeoff between advice and competitive ratio in **ONE-WAY TRADING**.

References

1. Kfir Barhum, Hans-Joachim Böckenhauer, Michal Forisek, Heidi Gebauer, Juraj Hromkovič, Sacha Krug, Jasmin Smula, and Björn Steffen. On the power of advice and randomization for the disjoint path allocation problem. In *Proc. of the 40th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2014)*, volume 8327 *Lecture Notes in Computer Science*, pages 89–101. Springer-Verlag 2014.
2. Hans-Joachim Böckenhauer, Juraj Hromkovič, Dennis Komm, Sacha Krug, Jasmin Smula, and Andreas Sprock. The string guessing problem as a method to prove lower bounds on the advice complexity. *Theoretical Computer Science*, 554:95–108, Elsevier Science Publishers, 2014.
3. Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Kráľovič, and Richard Kráľovič. On the advice complexity of the k -server problem. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Proc. of the 38th international colloquium on Automata, languages and programming (ICALP 2011)*, volume 6755 of *Lecture notes in computer science*, pages 207–218. Springer-Verlag 2011.
4. Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Kráľovič, Richard Kráľovič, and Tobias Mömke. On the advice complexity of online problems. In Yingfei Dong, Ding-Zhu Du, and Oscar H. Ibarra, editors, *Proc. of the 20th International Symposium on Algorithms and Computation (ISAAC 2009)*, volume 5878 of *Lecture Notes in Computer Science*, pages 331–340. Springer-Verlag 2009.

5. Hans-Joachim Böckenhauer, Dennis Komm, Richard Kráľovič, and Peter Rossmanith. The online knapsack problem: Advice and randomization. *Theoretical Computer Science*, 527:61–72, 2014.
6. Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press 1998.
7. Joan Boyar, Kim S. Larsen, and Abyayananda Maiti. A comparison of performance measures via online search. *Proc. of the Joint International Conference on Frontiers in Algorithmics and Algorithmic Aspects in Information and Management (FAW-AAIM 2012)*, pages 303–314. Springer-Verlag 2012.
8. Joan Boyar, Lene M. Favrholdt, Christian Kudahl, and Jesper W. Mikkelsen. Advice complexity for a class of online problems. In *Proceedings of the 32nd Symposium on Theoretical Aspects of Computer Science (STACS 2015)*, volume 30 of *Leibniz International Proceedings in Informatics*, pages 116–129. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015.
9. Stefan Dobrev, Rastislav Kráľovič, and Dana Pardubská. How much information about the future is needed? In *Proc. of the 34th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2008)*, volume 4910 of *Lecture Notes in Computer Science*, pages 247–258. Springer-Verlag 2008.
10. Ran El-Yaniv, Amos Fiat, Richard Karp, and G. Turpin. Optimal search and one-way trading online algorithms. *Algorithmica*, 30:101–139, 2001.
11. Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. *Theoretical Computer Science*, 412(24):2642–2656, 2011.
12. Sushmita Gupta, Shahin Kamali, and Alejandro López-Ortiz. On advice complexity of the k -server problem under sparse metrics. In *Proc. of the 20th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2013)*, volume 8179 of *Lecture Notes in Computer Science*, pages 55–67. Springer-Verlag 2013.
13. Juraj Hromkovič, Rastislav Kráľovič, and Richard Kráľovič. Information complexity of online problems. In *Proc. of the 35th International Symposium on Mathematical Foundations of Computer Science (MFCS 2010)*, volume 6281 of *Lecture Notes in Computer Science*, pages 24–36. Springer-Verlag 2010.
14. Dennis Komm and Richard Kráľovič. Advice complexity and barely random algorithms. *Theoretical Informatics and Applications (RAIRO)*, 45(2), 249–267, 2011.
15. Julian Lorenz, Konstantinos Panagiotou, and Angelika Steger. Optimal algorithms for k -search with application in option pricing. *Algorithmica*, 55(2):311–328, 2009.
16. Jesper Mikkelsen. Randomization can be as helpful as a glimpse of the future in online computation. *CoRR*, abs/1511.05886, 2015.
17. Marc P. Renault and Adi Rosén. On online algorithms with advice for the k -server problem. In *9th International Workshop on Approximation and Online Algorithms (WAOA 2011)*, volume 7164 of *Lecture Notes in Computer Science*, pages 198–210. Springer-Verlag, 2011.
18. Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.
19. Björn Steffen. *Advice Complexity of Online Graph Problems*. PhD thesis, ETH Zurich, 2014.
20. Yinfeng Xu, Wenming Zhang, and Feifeng Zheng. Optimal algorithms for the online time series search problem. *Theoretical Computer Science*, 412(3):192–197, 2011.
21. Wenming Zhang, Yinfeng Xu, Feifeng Zheng, and Ming Liu. Online algorithms for the general k -search problem. *Information Processing Letters*, 111(14):678–682, 2011.

22. Wenming Zhang, E Zhang, and Feifeng Zheng. Online (J, K) -search problem and its competitive analysis. *Theoretical Computer Science*, 593:139–145, 2015.